(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 March 2002 (28.03.2002)

PCT

(10) International Publication Number
WO 02/25504 A2

(51) International Patent Classification⁷: G06F 17/50

(21) International Application Number: PCT/US01/29140

(22) International Filing Date: 20 September 2001 (20.09.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/234,206 20 September 2000 (20.09.2000) US

(71) Applicants and
(72) Inventors: **LOBANOV, Victor, S.** [RU/US]; 1002 Darby Drive, Yardley, PA 19067 (US). **AGRAFIOTIS, Dimitris, K.** [US/US]; 660 Perimeter Drive, Downingtown, PA 19335 (US). **SALEMME, Francis, R.** [US/US]; 1970 Timber Lakes Drive, Yardley, PA 19067 (US).

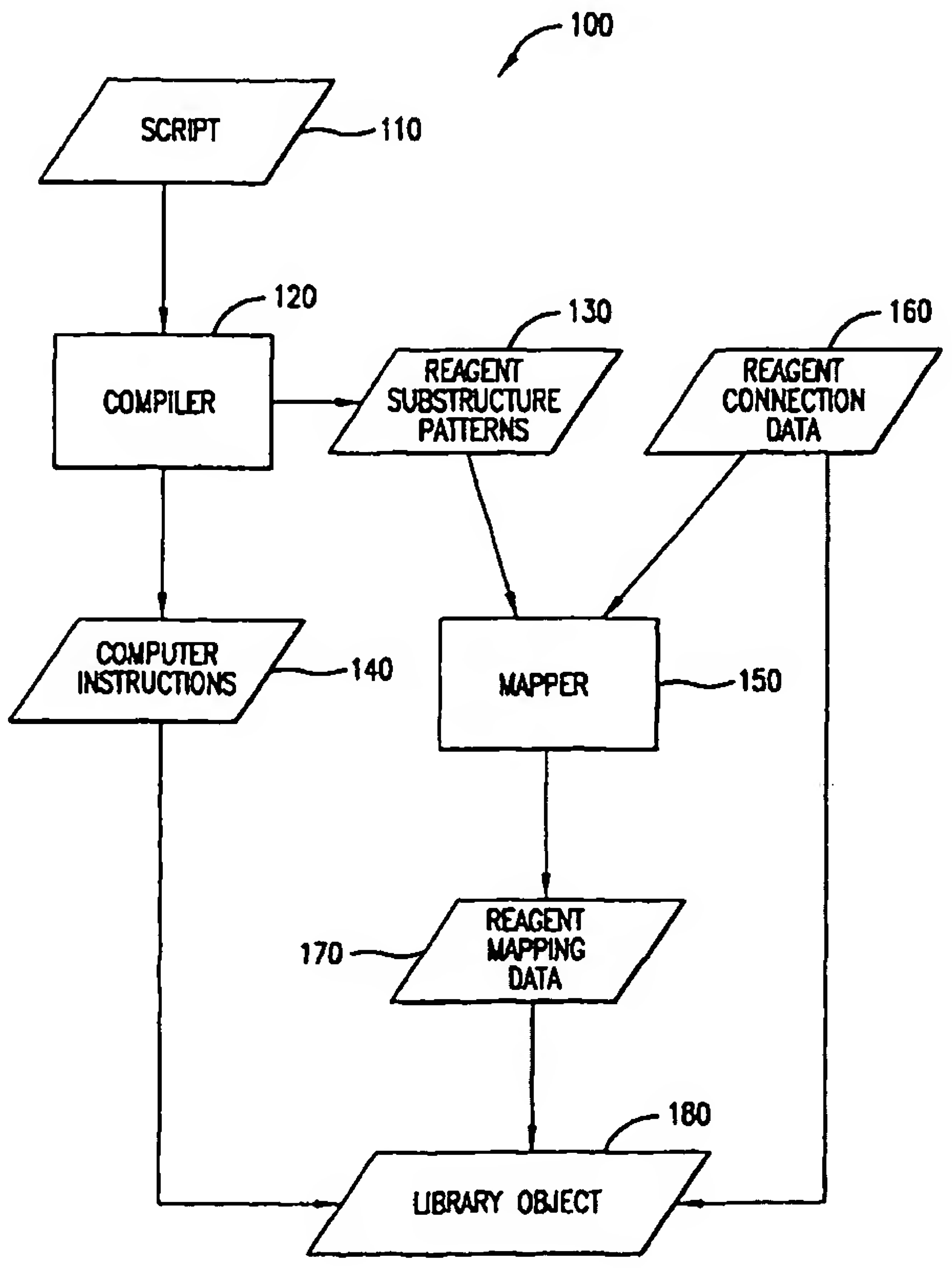
(74) Agents: **LEE, Michael, Q.** et al.; Sterne, Kessler, Goldstein, & Fox P.L.L.C., Suite 600, 1100 New York Avenue, N.W., Washington, DC 20005-3934 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European

[Continued on next page]

(54) Title: METHOD, SYSTEM, AND COMPUTER PROGRAM PRODUCT FOR ENCODING AND BUILDING PRODUCTS OF A VIRTUAL COMBINATORIAL LIBRARY



(57) **Abstract:** The present invention provides a method, system, and computer program product for encoding and building products of a virtual combinatorial library. A chemical reaction and reagent data for forming products of the virtual combinatorial library are encoded in a computer readable form. A compiler then operates on the encoded information and generates reagent mapping data. The compiler compiles the encoded chemical reaction to generate computer instructions that control the operation of a processor. A compact data structure containing data is generated and stored in a memory. This data structure is then used to gain immediate access to any of the products of the virtual combinatorial library.

WO 02/25504 A2

patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

METHOD, SYSTEM, AND COMPUTER PROGRAM PRODUCT FOR ENCODING AND BUILDING PRODUCTS OF A VIRTUAL COMBINATORIAL LIBRARY

FIELD OF THE INVENTION

The present invention relates to combinatorial chemistry. More particularly, it relates to virtual combinatorial libraries used in computer aided molecular design.

BACKGROUND OF THE INVENTION

Among the tools available to a medicinal chemist, combinatorial chemistry is one of the most powerful and best suited for exploring chemical space in search of new drug leads. Combinatorial chemistry provides access to millions of novel compounds from a limited number of building blocks using synthetic procedures that work reliably across a wide range of starting materials.

A virtual combinatorial library is a collection of chemical compounds or products, in electronic form, generated by combining a number of chemical building blocks such as reagents. For example, a polypeptide virtual combinatorial library can be formed by combining a set of chemical building blocks called amino acids, in electronic form, in every possible or nearly every possible way for a given compound length (i.e., the number of amino acids in a polypeptide compound).

Generally speaking, there are two kinds of virtual combinatorial libraries that can be formed: a viable library and an accessible library. A viable library is relatively small in size. It is assembled from readily available reagents that have been filtered, for example, by a medicinal chemist. A viable library will often have a physical counterpart. An accessible library, on the other hand, is relatively large in size. It can encompass millions or billions of products. An accessible library will typically include all possible reagents that are in principle compatible with a particular chemical reaction scheme. Typically, an accessible library is so large that it can never be physically synthesized in its entirety. Thus, in many cases, appropriate selection techniques must be applied to an accessible library in order to identify a subset of compounds or products for physical synthesis and biological testing. In order to take advantage of robotic hardware, minimize the number of reagents, and simplify the logistical aspects of a chemical experiment, physical libraries are almost invariably synthesized in the form of arrays, which represent the products derived by combining a given subset of reagents in all possible combinations as prescribed by the reaction scheme.

Depending on their use, virtual combinatorial libraries are divided into two main categories: (1) focused or directed libraries, which are biased against a specific target, structural class, or known pharmacophore; and (2) exploratory or probe libraries, which are target-independent and are designed to span a wide range of physicochemical and structural characteristics. Focused libraries are typically designed to follow up on a known lead, optimize a set of properties, or validate some structure-activity hypothesis. Access to the chemical structures of the products is required in order to assess molecular similarity, predict biological activity, or estimate some other property of interest. In contrast, probe libraries explore chemical space in search of novel hits, and their design is based predominantly on molecular diversity. Although fairly diverse libraries can be built by selecting a diverse set of reagents, there is overwhelming evidence (see V. J. Gillet et al., The effectiveness of reactant pools for generating structurally-diverse combinatorial libraries, *J. Chem. Inf. Comput. Sci.*, 1997, *37*, 731-740; and E. A. Jamois et al., Evaluation of reagent-based and product-based strategies in the design of combinatorial library subsets, *J. Chem. Inf. Comput. Sci.*, 2000, *40*, 63-70), but not conclusive evidence (see A. Linusson et al., Statistical Molecular Design of Building Blocks for Combinatorial Chemistry, *J. Med. Chem.*, 2000, *43*, 1320-1328; and E. J. Martin et al., Oriented Substituent Pharmacophore PropErtY Space (OSPPREYS): A substituent-based calculation that describes combinatorial library products better than the corresponding product-based calculation, *J. Mol. Graphics Modell.*, 2000, *18*, 383-403), which suggests that product-based designs are substantially better.

Experience suggests that selections based exclusively on molecular diversity tend to include "extreme" reagents, which can increase cost, cause delays due to limited availability, lead to unforeseen synthetic problems, and produce unusual compounds of limited pharmaceutical interest. The hit rate achieved with such libraries has proven disappointingly low (see A. R. Leach and M. M. Hann, The *in silico* world of virtual libraries, *Drug Discovery Today*, 2000, *5*, 326-336), and the compounds often exhibit unfavorable biological properties that could potentially result in ADME liabilities (see C. A. Lipinski et al., Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug Deliv. Rev.* 1997, *23*, 3-25; and D. N. Rassokhin and D. K. Agrafiotis, Kolmogorov-Smirnov statistic and its application in library design, *J. Mol. Graphics Modell.*, 2000, *18(4-5)*, 370-384). Thus, the focus in the design of probe libraries has began to shift from pure diversity to chemical feasibility, availability of monomers, and drug likeness (see D. N. Rassokhin and D. K. Agrafiotis, Kolmogorov-Smirnov statistic and its application in library design, *J. Mol. Graphics Modell.*, 2000, *18(4-5)*, 370-384; A. R. Leach and M. M. Hann, The *in silico* world of virtual libraries, *Drug Discovery Today*, 2000, *5*, 326-336; J. Sadowski and H. Kubinyi, A scoring scheme for distinguishing between drugs and non-drugs. *J. Med. Chem.*, 1998, *41*, 3325-3329; Ajay et al., Can we learn to distinguish between "drug-like" and "nondrug-like" molecules?, *J. Med. Chem.*, 1998, *41*, 3314-3324; and J. Wang and K. Ramnarayan, Toward designing drug-like libraries: a novel computational approach for prediction of drug feasibility of compounds, *J. Comb. Chem.*, 1999, *1*, 524-533).

Creating designs that combine molecular diversity or similarity with desired property profiles and drug likeness requires the use of optimization techniques such as simulated annealing (see D. K. Agrafiotis, Stochastic algorithms for maximizing molecular diversity, *J. Chem. Inf. Comput. Sci.*, 1997, *37*, 841-851; D. K. Agrafiotis, On the use of information theory for assessing molecular diversity. *J. Chem. Inf. Comput. Sci.*, 1997, *37(3)*, 576-580; D. K. Agrafiotis and V. S. Lobanov, An efficient implementation of distance-based diversity metrics based on k-d trees, *J. Chem. Inf. Comput. Sci.*, 1999, *39(1)*, 51-58; M. Hassan et al., Optimization and visualization of molecular diversity of combinatorial libraries, *J. Comput. Aided. Mol. Des.*, 1996, *2*, 64-74; and A. C. Good and R. A. Lewis, New methodology for profiling combinatorial libraries and screening sets: cleaning up the design process with HARPick, *J. Med. Chem.*, 1997, *40*, 3926-3936) or genetic algorithms (see United States Patents 5,463,564; 5,574,656; 5,684,711; and 5,901,069 to D. K. Agrafiotis et al.; R. D. Brown and Y. C. Martin, Designing combinatorial library mixtures using a genetic algorithm, *J. Med. Chem.*, 1997, *40*, 2304-2313; and V. J. Gillet et al., Selecting combinatorial libraries to optimize diversity and physical properties, *J. Chem. Inf. Comput. Sci.*, 1999, *39*, 169-177) and access to the properties of the individual products. To that end, *in silico* enumeration or virtual library generation becomes an essential part of the design process.

Despite advances in the processing speed and storage capacity of modern computers, there are many combinatorial libraries that defy enumeration. Enumeration, or product expansion, refers to the translation of a library into a database containing connection tables for the products of the library. For example, it is easy to imagine a combinatorial library containing 10^{12} compounds (see R. D. Cramer et al., Virtual compound libraries: a new approach to decision making in molecular discovery research. *J. Chem. Inf. Comput. Sci.* 1998, *38*, 1010-1023), which would require over three years to enumerate at a rate of 10,000 structures per second. Since most of the descriptors that are typically employed in diversity profiling, similarity searching and QSAR are calculated at a much slower rate, an exhaustive analysis of such a library would be impossible. Hence, there is a need for virtual library enumeration and analysis techniques that are scalable and that can be applied to massive virtual libraries containing hundreds of millions of compounds.

SUMMARY OF THE INVENTION

The present invention provides a method, system, and computer program product for encoding and building products of a virtual combinatorial library. As described herein, the invention involves a pre-calculation or encoding stage in which data and computer instructions needed to build products of a virtual combinatorial library are generated, compiled, and stored in a compact data structure for subsequent retrieval. This stage of the invention eliminates any need to fully enumerate the virtual combinatorial library whenever a product is needed. The invention also involves a real-time or building stage, in which the data and computer instruction of the stored data structure are accessed and used, for example, to quickly build or generate product connection tables for selected product of the library on an as needed basis.

As described herein, during the encoding stage of embodiments of the invention at least one chemical transformation for generating product connection data from reagent connection data and one or more reagent substructure patterns involved in forming the products of the virtual combinatorial library are encoded in a computer readable form (e.g., a scripting language). A compiler operates on the encoded information and generates reagent mapping data. The reagent mapping data is generated from the one or more reagent substructure patterns and reagent connection data for a set of reagents from which the products of the virtual combinatorial library are formed. In an embodiment, the reagent mapping data encodes how an atom or group of atoms of the one or more reagent substructure patterns is mapped to an atom or group of atoms of a reagent molecule. The compiler compiles the encoded at least one chemical transformation to generate computer instructions that can control the operation of a processor. A library object containing the compiled computer instructions, the generated reagent mapping data, and the reagent connection data for the set of reagents is then generated and stored in a memory.

During the building stage of embodiments of the invention, a builder is used to generate product connection data for the products of the virtual combinatorial library. The builder uses the compiled computer instructions stored as a part of the library object. The builder operates on reagent mapping data and reagent connection data retrieved from the library object. In an embodiment, the reagent mapping data is stored as a plurality of reaction maps, and the reagent connection data for the set of reagents is stored as a plurality of reagent connection tables. In an embodiment, the output of the builder is a product connection table for each product built.

In an embodiment, data needed to build a particular product is retrieved using a product identification number. In another embodiment, data needed to build a particular product is retrieved using an identification number associated with one or more reagents used to form the particular product.

Further embodiments, features, and advantages of the present invention, as well as the structure and operation of the various embodiments of the present invention, are described in detail below with reference to the accompanying figures.

BRIEF DESCRIPTION OF THE DRAWINGS/FIGURES

The present invention is described with reference to the accompanying drawings wherein:

FIG. 1 illustrates a flowchart of an embodiment of the invention for encoding products of a virtual combinatorial library;

FIG. 2 illustrates a flowchart of an embodiment of the invention for building products of a virtual combinatorial library;

FIGs. 3A-B illustrate a flowchart of a method for encoding and building products of a virtual combinatorial library according to an embodiment of the invention;

FIG. 4 illustrates a flowchart of a method for building products of a virtual combinatorial library according to an embodiment of the invention;

FIG. 5 illustrates a library object according to an embodiment of the invention;

FIG. 6 illustrates how a reaction map is generated according to an embodiment of the invention;

FIG. 7 illustrates an example product connection table according to an embodiment of the invention;

FIG. 8 illustrates example script for generating a virtual combinatorial library based on the reductive amination reaction according to an embodiment of the invention;

FIG. 9 illustrates frequently used operators (instructions) according to an embodiment of the invention;

FIG. 10 illustrates an example encoding of a stereochemical reaction according to an embodiment of the invention;

FIG. 11 illustrates the encoded reaction of FIG. 10;

FIG. 12 illustrates an example encoding of a stereochemical reaction according to an embodiment of the invention;

FIG. 13 illustrates the encoded reaction of FIG. 12;

FIG. 14 illustrates an example encoding of a stereochemical reaction according to an embodiment of the invention;

FIG. 15 illustrates the encoded reaction of FIG. 14;

FIG. 16 illustrates an example encoding of a stereochemical reaction according to an embodiment of the invention;

FIG. 17 illustrates the encoded reaction of FIG. 16; and

FIG. 18 illustrates an exemplary computing environment according to an embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

Embodiments of the present invention are now described with references to the figures, where like reference numbers indicate identical or functionally similar elements. Also in the figures, the left most digit(s) of each reference number corresponds to the figure in which the reference number is first used. While specific configurations and arrangements are discussed, it should be understood that this is done for illustrative purposes only. One skilled in the relevant art will recognize that other configurations and arrangements can be used without departing from the spirit and scope of the invention. It will also be apparent to one skilled in the relevant art(s) that this invention can also be employed in a variety of other devices and applications.

Overview of the Invention

The present invention provides a method, system, and computer program product for encoding and building products of a virtual combinatorial library. As described herein, the invention involves a pre-calculation or encoding stage in which data and computer instructions needed to build products of a virtual combinatorial library are generated, compiled, and stored in a compact data structure for subsequent retrieval. This stage of the invention eliminates any need to fully enumerate the virtual combinatorial library whenever a product is needed. The invention also involves a real-time or building stage, in which the data and computer instruction of the stored data structure are accessed and used, for example, to quickly build or generate product connection tables for selected product of the library on an as needed basis.

In operation during the encoding stage, at least one chemical transformation for generating product connection data from reagent connection data and reagent substructure patterns involved in forming the products of the virtual combinatorial library are encoded in a computer readable form (e.g., a scripting language). A compiler then operates on the encoded information and generates reagent mapping data. In an embodiment, the reagent mapping data encodes how an atom or group of atoms of the reagent substructure patterns are mapped to an atom or group of atoms of a reagent molecule. The compiler also compiles the encoded chemical transformation to generate computer instructions that can control the operation of a processor. A data structure, referred to herein as a library object, is then generated. This library object can be stored in a memory (e.g., on a computer disk) and used at a later time to build any or all of the products of the virtual combinatorial library. In an embodiment, the library object includes compiled computer instructions, reagent mapping data, and reagent connection data.

As described herein, during the building stage of the invention, a library object according to the invention can be used to gain immediate access to any of the products of the virtual combinatorial library. In an embodiment, a builder according to the invention uses the compiled computer instructions of a library object, and operates on reagent mapping data and reagent connection data retrieved from the library object to generate products or product connection data. In an embodiment, the output of the builder is a product connection table for each product built.

Embodiments of the Invention

FIGs. 1-4 illustrate embodiments of the invention for encoding and/or building products of a virtual combinatorial library.

FIG. 1 illustrates a flowchart of an embodiment 100 for encoding the products of a virtual combinatorial library in a compact data structure or library object 180 according to the invention. As described herein, embodiment 100 facilitates fast, "on-demand" enumeration or building of combinatorial products. As described in detail below, method 100 comprises three operation stages or steps. A person skilled in the relevant art will understand how to implement embodiment 100 based on the description of the invention herein.

Prior to step one of method 100, a chemical reaction or chemical transformation used to combine reagents and form products is encoded in a computer readable form or script 110. In an embodiment, a reaction scripting language (RSL) or scripting language is used to facilitate the encoding of chemical transformations. An RSL according to an embodiment of the invention is described below in the Library Construction section. Advantages of using an RSL according to the invention include versatility, speed, and scalability.

In stage or step one of embodiment 100, a compiler 120 is used to operate on script 110 and generate reagent substructure patterns 130. FIG. 6 illustrates an example reagent substructure pattern 610. In an embodiment, SMARTS notation (see C. A. James et al., Daylight Theory Manual Daylight 4.71, Daylight Chemical Information Systems, Inc., 2000, http://www.daylight.com/dayhtml/doc/theory/theory.toc.html) is used to encode the reagent substructural patterns 130 that are involved in the encoded chemical transformations and that must be present in order for reagents selected from a set of reagents to undergo a reaction. Each reagent can be defined using multiple patterns. The order in which the multiple patterns are defined in script 110 specifies the relative reactivity of the respective functional groups. For example, for the amination library of FIG. 8, lines 5 and 6 specify that both primary and secondary amines can react with an aldehyde. This feature of the invention is further described below in the Library Construction section.

In stage or step one of embodiment 100, compiler 120 is also used to operate on script 110 and generate computer instructions 140. As described in more below in the Library construction section and elsewhere herein, computer instructions 140 are compiled computer instructions or computer logic for controlling the operation of a processor. FIG. 5 shows illustrative compiled computer instructions 540. Computer instructions 140 are used to control a processor and thereby generate product connection data from reagent connection data and reagent mapping data (e.g., reaction maps) according to the invention. As illustrated in FIG. 5, computer instructions 140 form a part of library object 180.

In stage or step two of embodiment 100, a mapper 150 operates on reagent substructure patterns 130 and reagent connection data 160 to generate reagent mapping data 170. In an embodiment, reagent mapping data 170 encodes how an atom or group of atoms of a reagent substructure pattern is mapped to an atom or group of atoms of a reagent molecule. FIG. 6 illustrates this stage or step of embodiment 100.

As can be seen in FIG. 6, in an embodiment, reagent connection data 620 is combined with reagent substructure pattern 610 to form a reaction map 550 according to the invention. Reaction map 550 is one form in which reaction data 170 can be stored for later retrieval. Reaction map 550 encodes data identifying that atom or node zero of reagent substructure pattern 610 maps to atom or node six of reagent connection data 620. Node six is labeled by number 624 in FIG. 6. Reaction map 550 also encodes data identifying that atom or node one of reagent substructure pattern 610 maps to atom or node seven of reagent connection data 620. Node seven is labeled by number 626 in FIG. 6. Data regarding atom or node five, labeled by number 622, is not needed in order to build products, and thus does not form a part of the data encoded by reaction map 550.

As will be known to a person skilled in the relevant art, reagent connection data 620 can be stored in the form of a reagent connection table. An example reagent connection table 560 is illustrated in FIG. 5.

In the third stage or step of embodiment 100, computer instructions 140, reagent connection data 160, and reagent mapping data 170 are combined to form library object 180. This is illustrated in FIG. 5. As illustrated in FIG. 5, in an embodiment, reagent mapping data 170 is stored as a number of reaction maps 550 and reagent connection data 160 is stored as a plurality of connection tables 560. In other embodiments, other structures may be used to store this information as would be known to a person skilled in the relevant art. For the connection table shown in FIG. 5, illustrative table entries indicate the number of bonds connecting particular atoms or groups of atoms.

As noted above, further features of embodiment 100 are described elsewhere herein. As will be understood by a person skilled in the relevant art given the description herein, library object 180 can be used to build products (e.g., generate product connection data) of a virtual combinatorial library "on-demand."

FIG. 2 illustrates an embodiment 200 for generating products or product connection data 230 according to the invention. Embodiment 200 involves using a builder 220 to operate on library object 180 and generate product connection data 230. The particular product connection data 230 generated by embodiment 200 is determined using an identification number 210.

As described herein, builder 220 is used to generate product connection data for the products of a virtual combinatorial library. Builder 220 uses compiled computer instructions 140 stored as a part of library object 180. Builder 220 operates on reagent mapping data 170 and reagent connection data 160 retrieved from library object 180. In an embodiment, the reagent mapping data 170 of library object 180 is stored as a plurality of reaction maps 550, and the reagent connection data 160 for a set of reagents is stored as a plurality of reagent connection tables 560. In an embodiment, the output of the builder is a product connection table for each product built. This is illustrated in FIG. 7.

As shown in FIG. 7, a product connection table 700 can be built from two reagent connection tables 702 and 704. Reagent connection table 702 stores connection data for a molecule A. Reagent connection table 704 stores connection data for a molecule B. In order to form product connection table 700, reagent connection tables 702 and 704 are combined as shown in FIG. 7 to form an extended connection table. As will be understood by a person skilled in the relevant art given the description herein, the combined connection table can be modified by adding and/or deleting bond entries, and by adding and/or deleting columns and rows that represent particular atoms or groups of atoms. These operations are controlled by computer instructions 140.

As described herein, in an embodiment of the invention computer instructions 140 provide explicit instructions that control how a processor is operated to assemble product molecule from reagents. For simplicity and speed, computer instructions 140 do not explicitly construct any intermediates that might be formed during a chemical reaction, but rather summarize the transformation of the input reagents directly into final products. This is further described below in the Library Construction section. As described herein, library object 180 includes all of the computer code and data needed to generate the products of a particular virtual combinatorial library.

As will be understood by a person skilled in the relevant art, data to build a product connection table is retrieved from library object 180 using, for example, a pointer to a memory location. In an embodiment, this pointer is identification number 210. In an embodiment, data needed to build a particular product is retrieved using a product identification number. In another embodiment, data needed to build a particular product is retrieved using an identification number associated with one or more reagents used to form the particular product. How to store and retrieve information from library object 180 using identification number 210 will be understood by a person skilled in the relevant art given the description of the invention herein.

Further features of embodiment 200 are also described elsewhere herein.

FIGs. 3A-B illustrate the steps of a computer method according to the invention for encoding and building products of a virtual combinatorial library. Method 300 can be implemented using the system and computer embodiments of the invention described herein. The features of method 300 are described both below with regard to the features of embodiments 100 and 200.

In step 310, at least one chemical transformation for generating product connection data 230 from reagent connection data 160 is encoded in computer readable form or script 110. As described above, in embodiments a chemical reaction or chemical transformation used to combine reagents and form products is encoded in a computer readable form. In an embodiment, a reaction scripting language (RSL) or scripting language is used to facilitate the encoding of chemical transformations. An RSL according to an embodiment of the invention is described below in the Library Construction section.

In step 320, at least one reagent substructure pattern 130 involved in forming the products of the virtual combinatorial library is encoded in computer readable form. This step is described in detail below in the Library Construction section.

In step 330, reagent mapping data 170 is generated from at least one reagent substructure pattern 130 (reaction map 550) and reagent connection data 160 for a set of reagents. FIG. 6 illustrates how this is performed. As can be seen in FIG. 6, in an embodiment, reagent connection data 620 is combined with reagent substructure pattern 610 by a mapper 150 to form a reaction map 550 according to the invention. Reaction map 550 is one form in which reaction data 170 can be stored for later retrieval.

In step 340, the encoded chemical transformation or transformations of step 310 is compiled into computer instructions 140. In an embodiment, compiler 120 is used to operate on script 110 and generate computer instructions 140. As described in more below in the Library construction section and elsewhere herein, computer instructions 140 are compiled computer instructions or computer logic for controlling the operation of a processor. FIG. 5 shows illustrative compiled computer instructions 540. Computer instructions 140 are used to control a processor and thereby generate product connection data from reagent connection data and reagent mapping data (e.g., reaction maps) according to the invention.

In step 350, a library object 180 according to the invention is generated. Library object 180 includes the compiled computer instructions 140 of step 340, the generated reagent mapping data 170 of step 330, and reagent connection data 160 for the set of reagents used to form the virtual combinatorial library. FIG. 5 illustrates an example of a library object 180 according to the invention.

In step 360, the library object is stored in a memory. The memory can be any memory, such as for example memory 1810 of FIG. 18. In an embodiment, the reagent connection data 160 for the set of reagents is stored as a plurality of reagent connection tables 560. In an embodiment of the invention, the reagent mapping data 170 is stored as a plurality of reaction maps 550.

In step 370, product connection data 230 is generated for a product of the virtual combinatorial library, using the compiled computer instructions 140 and reagent mapping data 170 and reagent connection data 160 retrieved from stored library object 180. In an embodiment, step 370 results in the generation of one or more product connection tables 700. In an embodiment, a build 220 is used to generate product connection data 230. The operation of step 370 is further described below in the Library Construction section.

As described herein, in an embodiment, at least one reaction map 550 and at least the reagent connection data 560 associated with the reagents is retrieved in step 370 and used to form a product of the virtual combinatorial library. In an embodiment, data from library object 180 is retrieved using a product identification number. In another embodiment, data is retrieved from the library object 180 using an identification number associated with at least one reagent. Further features of computer method 300 are described elsewhere herein.

FIG. 4 illustrates a computer method 400 for building products of a virtual combinatorial library according to an embodiment of the invention.

The products of the virtual combinatorial library generated by method 400 are formed in accordance with a chemical reaction and selected reagents.

In step 410, a library object 180 is stored in a memory. In an embodiment, the library object 180 includes compiled chemical transformation computer instructions 140 that generate product connection data 230 from reagent connection data. The library object also includes reagent mapping data 170, and reagent connection data 160 for the set of reagents. In some embodiments, other information may be included in library object 180. For example, library object may include reagent pricing data and/or reagent availability data. Other types of data that would be useful to a user of a virtual combinatorial library can be included in library object 180, which would be known to a person skilled in the relevant art.

In step 420, product connection data 230 is generated for a product of the combinatorial library using the compiled computer instructions 140, reagent mapping data 170, and reagent connection data 160 retrieved from the stored library object 180. How this is done is described above with regard to embodiment 200, and below in the Library Construction section.

Further embodiments, features, and advantages of the present invention, as well as the structure and operation of the various embodiments of the present invention, are described in detail below.

Library Construction

As stated above, a reaction scripting language (RSL) or scripting language is used in embodiments of the invention to facilitate the encoding of chemical transformations and the enumeration of virtual libraries. The advantages of using a RSL include versatility, speed, and scalability.

In an embodiment, an RSL is designed as an extension of the Tool Command Language (Tcl) (see J. Ousterhout, Tcl and the Tk Toolkit, Addison-Wesley, ISBN 0-201-63337-X, 1994), which has a fairly simple and human-readable syntax. Each combinatorial reaction is defined as a named Tcl procedure, thus providing a framework for creating libraries of common reaction schemes. RSL procedures are designed to be compilable into a sequence of parameterized function calls that are be executed in order to assemble the product structures. This facilitates fast, "on-demand" enumeration of combinatorial products.

When a reaction procedure is invoked, it generates a virtual combinatorial library from lists of reagents supplied in SD or SMILES format. In an embodiment, the generated library is stored in a compact form on disk and can be used at a later time for immediate access to any of the product structures. The names of the input reagent and output library files are passed as arguments to the Tcl reaction procedure. FIG. 8 illustrates an example of an RSL script according to an embodiment of the invention based on the reductive amination reaction.

Conceptually, an RSL procedure consists of three blocks: a definition block, an assembly instruction block and an execution trigger. The definition block defines the reagents and the product, and specifies the reactive patterns (reagent substructure patterns). The assembly instruction block provides explicit instructions on how to assemble the product molecule from the reagents, and what parts of the source molecules are eliminated in the process. For simplicity and speed, the assembly instructions do not explicitly construct any intermediates that might be formed during the reaction, but rather summarize the transformation of the input reagents directly into the final product. When the reaction script is executed, the assembly instructions are translated into a sequence of parameterized function calls. The assembly sequence is saved within the virtual library (library object), and every time a product needs to be assembled, this sequence is executed with the appropriate parameters. Lastly, the execution trigger is a statement that triggers the mapping of the reactive patterns onto the supplied sets of reagents, the compilation of the assembly instructions (computer instructions), and the storage of the virtual library into a file. Thus, our virtual library consists of structures of input reagents in their original form, maps of substructure patterns onto input reagents, and a sequence of compiled assembly instructions that should be executed in order to build the connection table of a product.

In an embodiment, RSL uses the SMARTS notation (see C. A. James et al., Daylight Theory Manual Daylight 4.71, Daylight Chemical Information

Systems, Inc., 2000, http://www.daylight.com/dayhtml/doc/theory/theory.toc.html) to encode the substructural patterns that are involved in the reaction and must be present in order for the reagent to undergo the reaction. Each reagent can be defined using multiple patterns, and the order in which they are defined specifies the relative reactivity of the respective functional groups. For example, in the amination library in FIG. 8, lines 5 and 6 specify that both primary and secondary amines can react with an aldehyde. However, if both a primary and a secondary amine are present in the same molecule, the main product will be formed from the more reactive primary amine. By defining the SMARTS pattern corresponding to the primary amine before the secondary amine, one can ensure that the proper products will be assembled. Although it is possible to match both primary and secondary amines with a single SMARTS pattern, it would not be possible to differentiate their reactivity.

Sometimes it is difficult, if not impossible, to write a single SMARTS string that will match only the reactive substructure pattern. In this case, one can specify the substructures that are not reactive. For example, a simple amine pattern "C[NH2]" will match an amide as well, which is not susceptible to reductive amination. One can either modify the amine pattern as "[CX4][NH2]" or define the amide substructure "C(=O)[NH2]" and designate it as non-reactive. When non-reactive patterns are present, the invention looks for an overlap between the matched reactive and non-reactive substructures, and if they have at least one atom in common the reactive structure will be invalidated.

In an embodiment, after the reacting substructures of the reagents are defined, the remaining code of the reaction script encodes the instructions for product assembly. Once the product is defined (line 5 in Fig. 8), its name becomes a Tcl command that supports a series of molecular operations. These operations include addition of previously defined reagents, removal of atoms, addition and removal of bonds, changing of the bond order, etc. A list of the most frequently used operators for an embodiment is given in FIG. 9. Note that with the exception of the "add" command, which instructs the program to add the connection table of the reagent to the connection table of the product, the remaining assembly instructions require specification of individual atoms affected by the instructions. Individual atoms participating in the chemical transformation are referred to by the respective reagent's name and by the zero-based indices of the matching atoms in the respective SMARTS pattern. For example, the nitrogen atom from an "amine" reagent defined with the SMARTS pattern "C[NH2]" is referred to as "amine:1". Since SMARTS strings are written in a single line, all atom specifications defined in a pattern can be unambiguously numbered from left to right as they appear in the pattern string. Note that hydrogen atoms are part of the atom specification in SMARTS and therefore cannot be individually addressed. (See FIG. 9.)

Since the reagent definition can include multiple SMARTS patterns, it is important that the atoms referenced in the assembly instructions have the same indices in every pattern. For instance, the nitrogen atom in both the primary and secondary amines should have the same index (e.g. 1) if a single assembly instruction is to apply to both of them. Fortunately, it is always possible to write SMARTS specifications in the desired order using "ring closures" (numbers), disconnections (dots) and recursive atom environments. Moreover, in most cases, SMARTS encoding lends itself naturally to this requirement since multiple patterns are typically used to define variations of the same functionality, such as primary and secondary amines.

In embodiments of the invention, most if not all of the assembly instructions are obvious (e.g., "insert bond," "remove atom," "remove bond," "set atom charge," "set bond order," etc.) Note that there is no instruction to insert a single atom since all the atoms of a product must come from the reagents in accordance with the mass preservation law. Special instructions are also provided to define the stereochemical outcome of reactions controlled by steric approach preferences. In RSL embodiment, the major product of a stereochemical reaction can be specified, for example, in two ways: (1) via the configuration of the nascent chiral center(s), and (2) via the stereochemical character of a bond during addition and elimination. FIGs. 10-17 illustrate some examples of stereochemistry encoding in an RSL embodiment.

The stereochemical configuration of a formed chiral center can be identified as "unspecified," "racemic" or "inverse." "Unspecified" indicates that the exact configuration of the products is unknown or irrelevant (default). "Racemic" indicates that both the R and S stereoisomers are formed in comparable quantities. "Inverse" exchanges one of the chiral center's substituents during the reaction and inverts its configuration. The R/S assignment of the chiral center is automatically determined based on the original configuration and the CIP priority of the new substituent. In general, the stereochemical configuration of an atom can be specified by listing its substituents in clockwise order and designating the last substituent as an up (in front of the plane) or a down (behind the plane) wedge. In this case, the R/S assignment is based on the CIP priorities and order of the substituents. Alternatively, the R/S configuration of the chiral center could be explicitly specified, but this option is rarely used since the label depends on the CIP priorities of the individual building blocks.

Stereochemical ambiguity also emerges when the reaction mechanism involves multiple centers. For example, dehydrohalogenation leads to double bond formation via an *anti* elimination pathway, whereby two substituents are removed from opposite sides of the reduced single bond. The resulting double bond can be *cis* or *trans* depending on the original configuration of the bonded atoms. In an embodiment, RSL defines two keywords, "syn_product" and "anti_product," to specify whether an addition or elimination reaction proceeds in a *syn* or *anti* manner. Note that it is not always possible to identify a single product using these keywords. For completeness, the configuration of the double bond can also be explicitly specified as E or Z.

Finally, the "enumerate" statement triggers the creation of the virtual library. Although semantically simple, this statement is the complicated in its implementation. For scalability, the enumeration of the products must be implicit and must circumvent the creation of a connection table or even a record for every product in the library. This objective is accomplished by dividing the enumeration process in two steps. During the first step, the reacting and interfering functionalities are identified by matching the corresponding SMARTS patterns, and any reagents that are not compatible with the reaction transform are eliminated from further processing. This step involves mostly substructure searching, and scales linearly with the number of reagents that make up the virtual library. The second step involves the generation of products and is delayed until a particular product is requested. That is, the construction of the connection table of a particular product occurs only when its structure is needed for display or evaluation, and in many cases this never happens. In the next section, several methods to analyze a virtual library are described that require the enumeration of only a minor fraction of its members (products).

In order to accelerate the "on-demand" assembly of products, the mappings of the reactive groups matched by the SMARTS patterns are stored within the virtual library along with the compiled sequence of assembly operators. Thus, when the structure of a product is needed, no time is spent on substructure searching or parsing assembly instructions. This design and the speed of the underlying foundation classes and molecular perception algorithms upon which the invention software is based, enable the construction of products at a rate of 10,000 structures per second on a 800 MHz Pentium III processor, including full perception of valence, rings and aromaticity.

The reaction scripting language of the invention differs from other reaction languages, such as SMIRKS (see C. A. James et al., Daylight Theory Manual Daylight 4.71. Daylight Chemical Information Systems, Inc., 2000, http://www.daylight.com/dayhtml/doc/theory/theory.toc.html), in that it is designed specifically for generating virtual combinatorial libraries and not for reaction database searching. Thus, the RSL of the invention is less cryptic, provides more flexibility in encoding reaction transformations, and can be stored in a compiled form to allow ultra-fast product enumeration. In addition, in the RSL of the invention, SMARTS patterns of the reacting functionalities are not restricted as in SMIRKS, where bond queries are not allowed and atomic expressions cannot contain queries if the bond order or connectivity change (see C. A. James et al., Daylight Theory Manual Daylight 4.71. Daylight Chemical Information Systems, Inc., 2000, http://www.daylight.com/dayhtml/doc/theory/theory.toc.html).

Computer and Computer Program Product Embodiments of the Invention

As will be understood by a person skilled in the relevant arts given the description herein, FIG. 18 shows an example computer system 1800 that supports implementation of the present invention. The present invention may be implemented using hardware, software, firmware, or a combination thereof. It may be implemented in a computer system or other processing system. The computer system 1800 includes one or more processors, such as processor 1804. The processor 1804 is connected to a communication infrastructure 1806 (e.g., a bus or network). Various software embodiments can be described in terms of this exemplary computer system. After reading this description, it will become apparent to a person skilled in the relevant art how to implement the invention using other computer systems and/or computer architectures.

Computer system 1800 also includes a main memory 1808, preferably random access memory (RAM), and may also include a secondary memory 1810. The secondary memory 1810 may include, for example, a hard disk drive 1812 and/or a removable storage drive 1814, representing a floppy disk drive, a magnetic tape drive, an optical disk drive, etc. The removable storage drive 1814 reads from and/or writes to a removable storage unit 1818 in a well-known manner. Removable storage unit 1818 represents a floppy disk, magnetic tape, optical disk, etc. As will be appreciated, the removable storage unit 1818 includes a computer usable storage medium having stored therein computer software and/or data. In an embodiment of the invention, removable storage unit 1818 can contain input data to be projected.

Secondary memory 1810 can also include other similar means for allowing computer programs or input data to be loaded into computer system 1800. Such means may include, for example, a removable storage unit 1822 and an interface 1820. Examples of such may include a program cartridge and cartridge interface (such as that found in video game devices), a removable memory chip (such as an EPROM, or PROM) and associated socket, and other removable storage units 1822 and interfaces 1820, which allow software and data to be transferred from the removable storage unit 1822 to computer system 1800.

Computer system 1800 may also include a communications interface 1824. Communications interface 1824 allows software and data to be transferred between computer system 1800 and external devices. Examples of communications interface 1824 may include a modem, a network interface (such as an Ethernet card), a communications port, a PCMCIA slot and card, etc. Software and data transferred via communications interface 1824 are in the form of signals 1828 which may be electronic, electromagnetic, optical or other signals capable of being received by communications interface 1824. These signals 1828 are provided to communications interface 1824 via a communications path (i.e., channel) 1826. This channel 1826 carries signals 1828 and may be implemented using wire or cable, fiber optics, a phone line, a cellular phone link, an RF link and other communications channels. In an embodiment of the invention, signals 1828 can include input data to be projected.

Computer programs (also called computer control logic) are stored in main memory 1808 and/or secondary memory 1810. Computer programs may also be received via communications interface 1824. Such computer programs, when executed, enable the computer system 1800 to perform the features of the present invention as discussed herein. In particular, the computer programs, when executed, enable the processor 1804 to perform the features of the present invention. Accordingly, such computer programs represent controllers of the computer system 1800.

Conclusion

While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example, and not limitation. It will be apparent to persons skilled in the relevant art that various changes in detail can be made therein without departing from the spirit and scope of the invention. Thus the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

WHAT IS CLAIMED IS:

1. A computer method for encoding and building products of a virtual combinatorial library, the products of the virtual combinatorial library being formed in accordance with a chemical reaction and selected reagents, the computer method comprising the steps of:

(1) encoding in computer readable form at least one chemical transformation for generating product connection data from reagent connection data;

(2) encoding in computer readable form at least one reagent substructure pattern involved in forming the products of the virtual combinatorial library;

(3) generating reagent mapping data from the at least one reagent substructure pattern and reagent connection data for a set of reagents;

(4) compiling into computer instructions the encoded at least one chemical transformation;

(5) generating a library object comprising the compiled computer instructions, the generated reagent mapping data, and the reagent connection data for the set of reagents;

(6) storing the library object in a memory; and

(7) generating product connection data, using the compiled computer instructions and reagent mapping data and reagent connection data retrieved from the stored library object, for at least one product of the virtual combinatorial library.

2. The method of claim 1, wherein step (3) comprises:

generating data that encodes how an atom of the at least one reagent substructure pattern is mapped to an atom of a reagent molecule.

3. The method of claim 1, wherein step (6) comprises:

storing the reagent connection data for the set of reagents as a plurality of reagent connection tables.

4. The method of claim 1, wherein step (6) comprises:
storing the reagent mapping data as a plurality of reaction maps.

5. The method of claim 1, wherein step (7) comprises:
retrieving at least one reaction map and at least the reagent connection data associated with the reagents used to form the at least one product.

6. The method of claim 1, wherein step (7) comprises:
retreiving data from the library object using a product identification number.

7. The method of claim 1, wherein step (7) comprises:
retreiving data from the library object using an identification number associated with at least one reagent.

8. The method of claim 1, wherein step (7) comprises:
generating a product connection table for the at least one product.

9. A computer method for building products of a virtual combinatorial library, the products of the virtual combinatorial library being formed in accordance with a chemical reaction and selected reagents, the computer method comprising the steps of:

(1) storing a library object in a memory, the library object comprising:

compiled chemical transformation computer instructions that generate product connection data from reagent connection data, reagent mapping data generated from at least one reagent substructure pattern and reagent connection data for a set of reagents, and

reagent connection data for the set of reagents; and

(2) generating product connection data, using the compiled computer instructions and reagent mapping data and reagent connection data retrieved from the stored library object, for at least one product of the virtual combinatorial library.

10. The method of claim 9, wherein step (1) comprises:

storing data that encodes how an atom of the at least one reagent substructure pattern is mapped to an atom of a reagent molecule.

11. The method of claim 9, wherein step (1) comprises:

storing the reagent connection data for the set of reagents as a plurality of reagent connection tables.

12. The method of claim 9, wherein step (1) comprises:

storing the reagent mapping data as a plurality of reaction maps.

13. The method of claim 9, wherein step (2) comprises:

retrieving at least one reaction map and at least the reagent connection data associated with the reagents used to form the at least one product.

14. The method of claim 9, wherein step (2) comprises:

retreiving data from the library object using a product identification number.

15. The method of claim 9, wherein step (2) comprises:

retreiving data from the library object using an identification number associated with at least one reagent.

16. The method of claim 9, wherein step (2) comprises:

generating a product connection table for the at least one product.

17. A computer program product for encoding and building products of a virtual combinatorial library, the products of the virtual combinatorial library being formed in accordance with a chemical reaction and selected reagents, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling a processor, the computer program logic comprising:

a procedure that enables said processor to encode in computer readable form at least one chemical transformation for generating product connection data from reagent connection data;

a procedure that enables said processor to encode in computer readable form at least one reagent substructure pattern involved in forming the products of the virtual combinatorial library;

a procedure that enables said processor to generate reagent mapping data from the at least one reagent substructure pattern and reagent connection data for a set of reagents;

a procedure that enables said processor to compile into computer instructions the encoded at least one chemical transformation;

a procedure that enables said processor to generate a library object comprising the compiled computer instructions, the generated reagent mapping data, and the reagent connection data for the set of reagents;

a procedure that enables said processor to store the library object in a memory; and

a procedure that enables said processor to generate product connection data, using the compiled computer instructions and reagent mapping data and reagent connection data retrieved from the stored library object, for at least one product of the virtual combinatorial library.

18. The computer program product of claim 17, wherein the generated reagent mapping data comprises:

data that encodes how an atom of the at least one reagent substructure pattern is mapped to an atom of a reagent molecule.

19. The computer program product of claim 17, wherein

the reagent connection data for the set of reagents is stored as a plurality of reagent connection tables.

20. The computer program product of claim 17, wherein

the reagent mapping data is stored as a plurality of reaction maps.

21. The computer program product of claim 17, wherein

data from the library object is retrieved using a product identification number.

22. The computer program product of claim 17, wherein

data from the library object is retrieved using an identification number associated with at least one reagent.

23. The computer program product of claim 17, wherein

a product connection table is generated for the at least one product.

24. A computer program product for building products of a virtual combinatorial library, the products of the virtual combinatorial library being formed in accordance with a chemical reaction and selected reagents, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling a processor, the computer program logic comprising:

a library object comprising:

compiled chemical transformation computer instructions that generate product connection data from reagent connection data,

reagent mapping data generated from at least one reagent substructure pattern and reagent connection data for a set of reagents, and

reagent connection data for the set of reagents; and

a procedure that enables said processor to generate product connection data, using the compiled computer instructions and reagent mapping data and reagent connection data retrieved from the library object, for at least one product of the virtual combinatorial library.

25. The computer program product of claim 24, wherein the generated reagent mapping data comprises:

data that encodes how an atom of the at least one reagent substructure pattern is mapped to an atom of a reagent molecule.

26. The computer program product of claim 24, wherein

the reagent connection data for the set of reagents is stored as a plurality of reagent connection tables.

27. The computer program product of claim 24, wherein

the reagent mapping data is stored as a plurality of reaction maps.

28. The computer program product of claim 24, wherein

data from the library object is retrieved using a product identification number.

29. The computer program product of claim 24, wherein

data from the library object is retrieved using an identification number associated with at least one reagent.

30. The computer program product of claim 24, wherein

a product connection table is generated for the at least one product.

31. A computer system for encoding and building products of a virtual combinatorial library, the products of the virtual combinatorial library being formed in accordance with a chemical reaction and selected reagents, the computer system comprising:

means for encoding in computer readable form at least one chemical transformation for generating product connection data from reagent connection data;

means for encoding in computer readable form at least one reagent substructure pattern involved in forming the products of the virtual combinatorial library;

means for generating reagent mapping data from the at least one reagent substructure pattern and reagent connection data for a set of reagents;

means for compiling into computer instructions the encoded at least one chemical transformation;

means for generating a library object comprising the compiled computer instructions, the generated reagent mapping data, and the reagent connection data for the set of reagents;

means for storing the library object in a memory; and

means for generating product connection data, using the compiled computer instructions and reagent mapping data and reagent connection data retrieved from the stored library object, for at least one product of the virtual combinatorial library.

32. A computer system for building products of a virtual combinatorial library, the products of the virtual combinatorial library being formed in accordance with a chemical reaction and selected reagents, the computer system comprising:

means for storing a library object in a memory, the library object comprising:

compiled chemical transformation computer instructions that generate product connection data from reagent connection data,

reagent mapping data generated from at least one reagent substructure pattern and reagent connection data for a set of reagents, and

reagent connection data for the set of reagents; and

means for generating product connection data, using the compiled computer instructions and reagent mapping data and reagent connection data retrieved from the stored library object, for at least one product of the virtual combinatorial library.

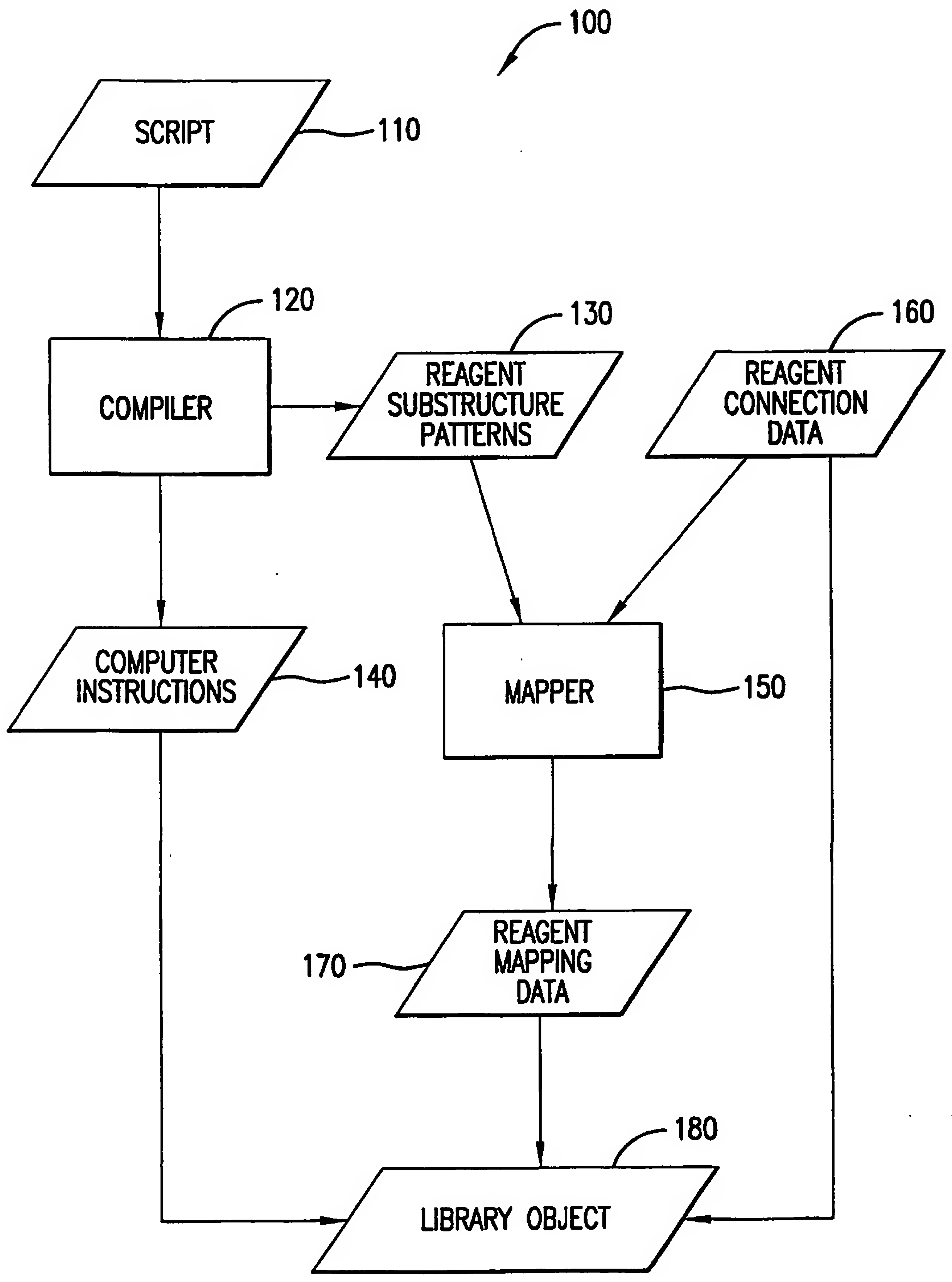

100
SCRIPT
110
120
COMPILER
130
REAGENT SUBSTRUCTURE PATTERNS
160
REAGENT CONNECTION DATA
COMPUTER INSTRUCTIONS
140
MAPPER
150
170
REAGENT MAPPING DATA
180
LIBRARY OBJECT

FIG. 1

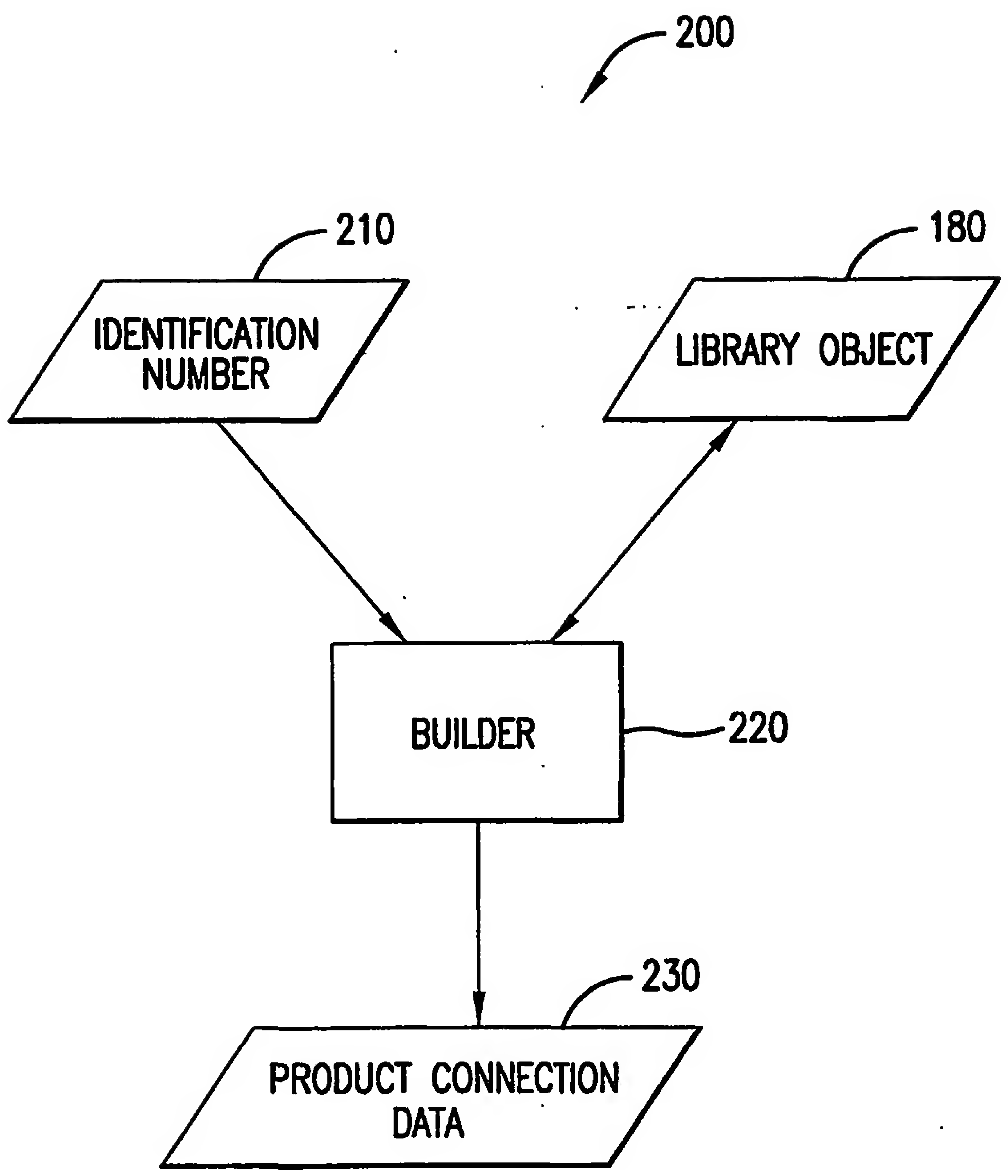

200
210
IDENTIFICATION NUMBER
180
LIBRARY OBJECT
BUILDER
220
230
PRODUCT CONNECTION DATA

FIG. 2

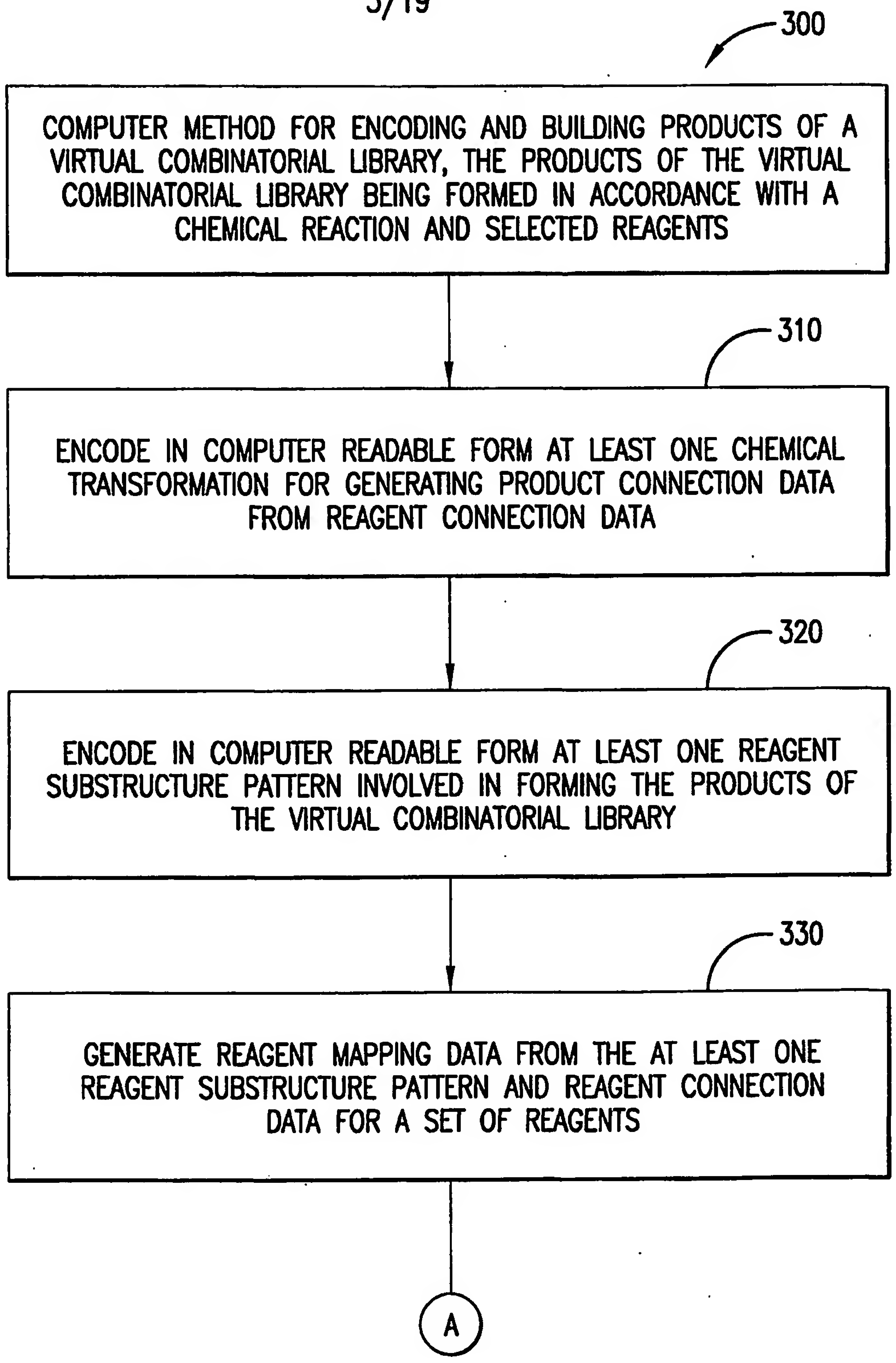

300
COMPUTER METHOD FOR ENCODING AND BUILDING PRODUCTS OF A VIRTUAL COMBINATORIAL LIBRARY, THE PRODUCTS OF THE VIRTUAL COMBINATORIAL LIBRARY BEING FORMED IN ACCORDANCE WITH A CHEMICAL REACTION AND SELECTED REAGENTS
310
ENCODE IN COMPUTER READABLE FORM AT LEAST ONE CHEMICAL TRANSFORMATION FOR GENERATING PRODUCT CONNECTION DATA FROM REAGENT CONNECTION DATA
320
ENCODE IN COMPUTER READABLE FORM AT LEAST ONE REAGENT SUBSTRUCTURE PATTERN INVOLVED IN FORMING THE PRODUCTS OF THE VIRTUAL COMBINATORIAL LIBRARY
330
GENERATE REAGENT MAPPING DATA FROM THE AT LEAST ONE REAGENT SUBSTRUCTURE PATTERN AND REAGENT CONNECTION DATA FOR A SET OF REAGENTS
A

FIG. 3A

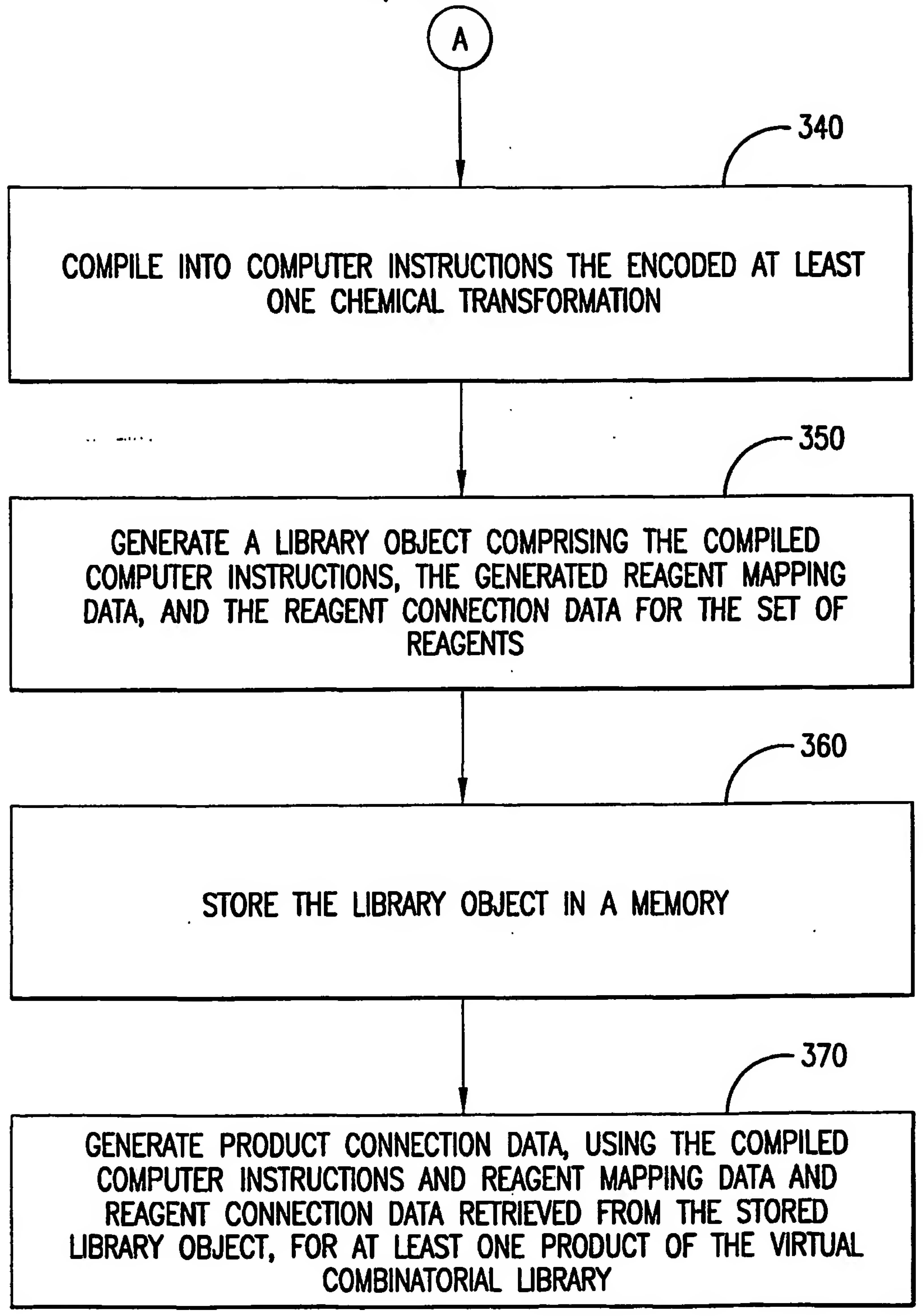
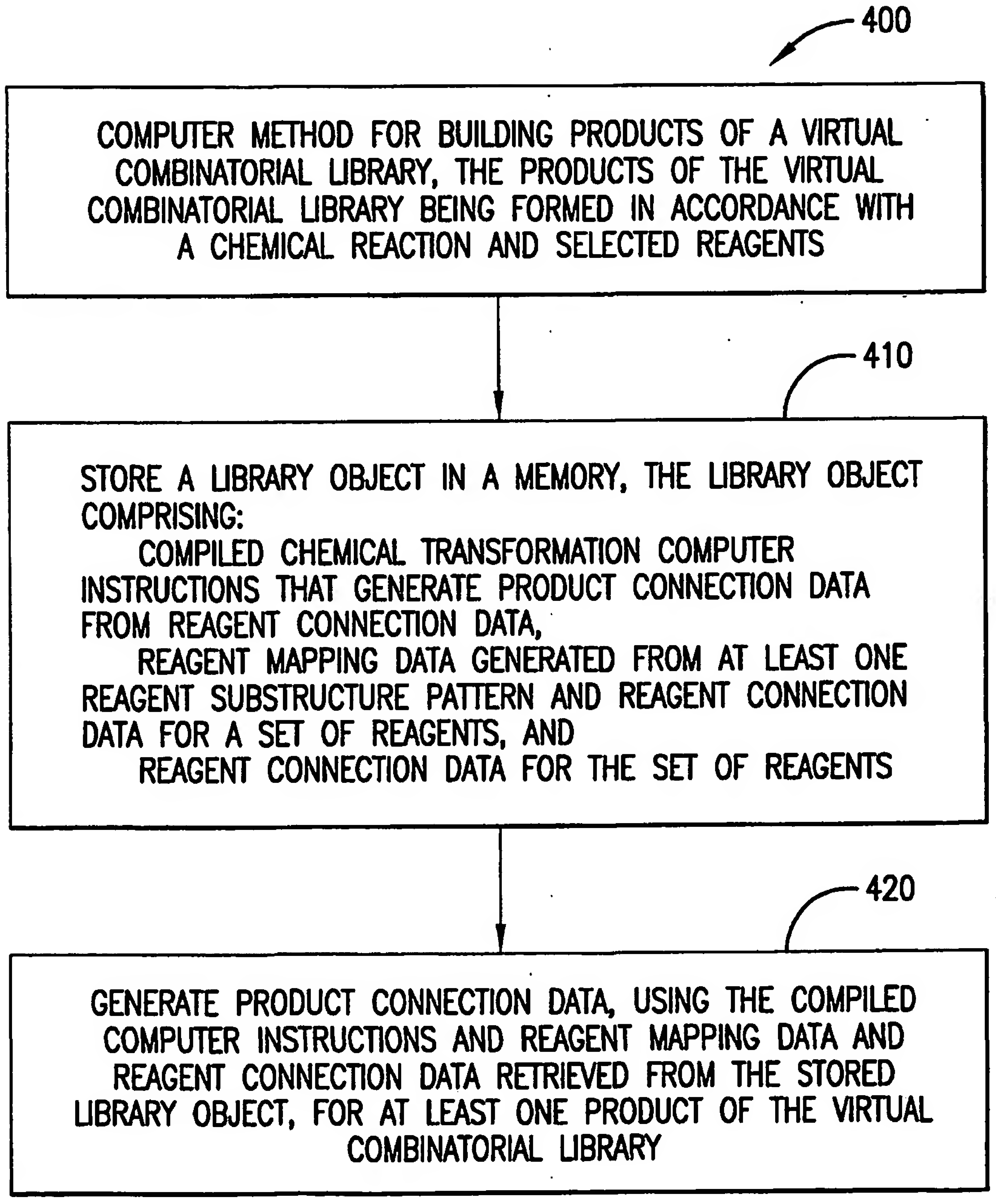

A
340
COMPILE INTO COMPUTER INSTRUCTIONS THE ENCODED AT LEAST ONE CHEMICAL TRANSFORMATION
350
GENERATE A LIBRARY OBJECT COMPRISING THE COMPILED COMPUTER INSTRUCTIONS, THE GENERATED REAGENT MAPPING DATA, AND THE REAGENT CONNECTION DATA FOR THE SET OF REAGENTS
360
STORE THE LIBRARY OBJECT IN A MEMORY
370
GENERATE PRODUCT CONNECTION DATA, USING THE COMPILED COMPUTER INSTRUCTIONS AND REAGENT MAPPING DATA AND REAGENT CONNECTION DATA RETRIEVED FROM THE STORED LIBRARY OBJECT, FOR AT LEAST ONE PRODUCT OF THE VIRTUAL COMBINATORIAL LIBRARY

FIG. 3B



COMPUTER METHOD FOR BUILDING PRODUCTS OF A VIRTUAL COMBINATORIAL LIBRARY, THE PRODUCTS OF THE VIRTUAL COMBINATORIAL LIBRARY BEING FORMED IN ACCORDANCE WITH A CHEMICAL REACTION AND SELECTED REAGENTS

STORE A LIBRARY OBJECT IN A MEMORY, THE LIBRARY OBJECT COMPRISING:
COMPILED CHEMICAL TRANSFORMATION COMPUTER INSTRUCTIONS THAT GENERATE PRODUCT CONNECTION DATA FROM REAGENT CONNECTION DATA,
REAGENT MAPPING DATA GENERATED FROM AT LEAST ONE REAGENT SUBSTRUCTURE PATTERN AND REAGENT CONNECTION DATA FOR A SET OF REAGENTS, AND
REAGENT CONNECTION DATA FOR THE SET OF REAGENTS

GENERATE PRODUCT CONNECTION DATA, USING THE COMPILED COMPUTER INSTRUCTIONS AND REAGENT MAPPING DATA AND REAGENT CONNECTION DATA RETRIEVED FROM THE STORED LIBRARY OBJECT, FOR AT LEAST ONE PRODUCT OF THE VIRTUAL COMBINATORIAL LIBRARY

FIG. 4

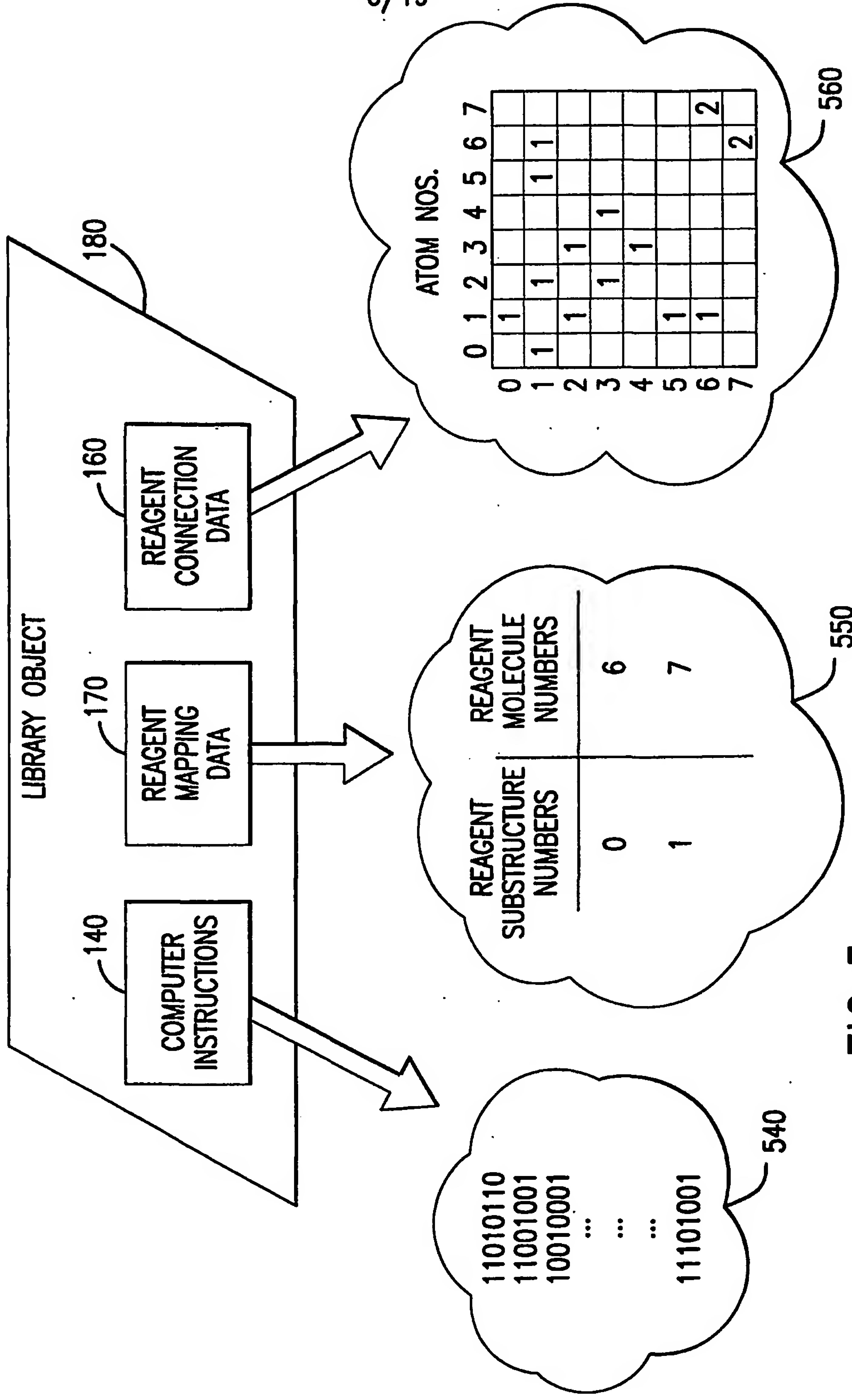

LIBRARY OBJECT
180
140
COMPUTER INSTRUCTIONS
170
REAGENT MAPPING DATA
160
REAGENT CONNECTION DATA
540
11010110
11001001
10010001
...
11101001
550
REAGENT SUBSTRUCTURE NUMBERS
0
1
REAGENT MOLECULE NUMBERS
6
7
560
ATOM NOS.
0 1 2 3 4 5 6 7
0: 1 at 1
1: 1 at 0, 2, 5, 6
2: 1 at 1, 3
3: 1 at 2, 4
4: 1 at 3
5: 1 at 1
6: 1 at 1; 2 at 7
7: 2 at 6

FIG. 5

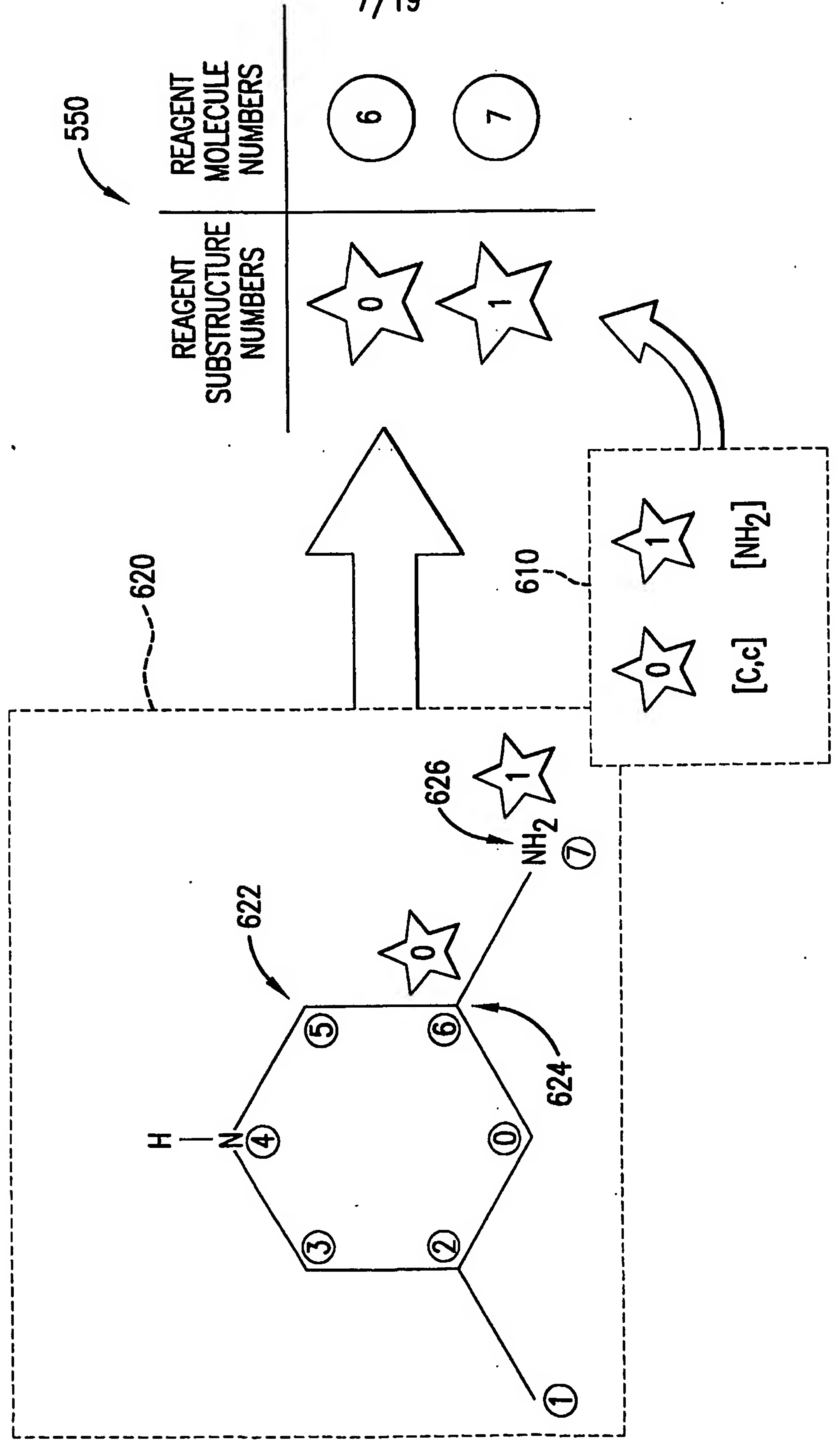

550
REAGENT MOLECULE NUMBERS
REAGENT SUBSTRUCTURE NUMBERS
6
7
0
1
620
610
0
1
[c,c]
[NH$_2$]
626
NH$_2$
7
622
0
5
6
624
H
N
4
0
3
2
1

FIG. 6

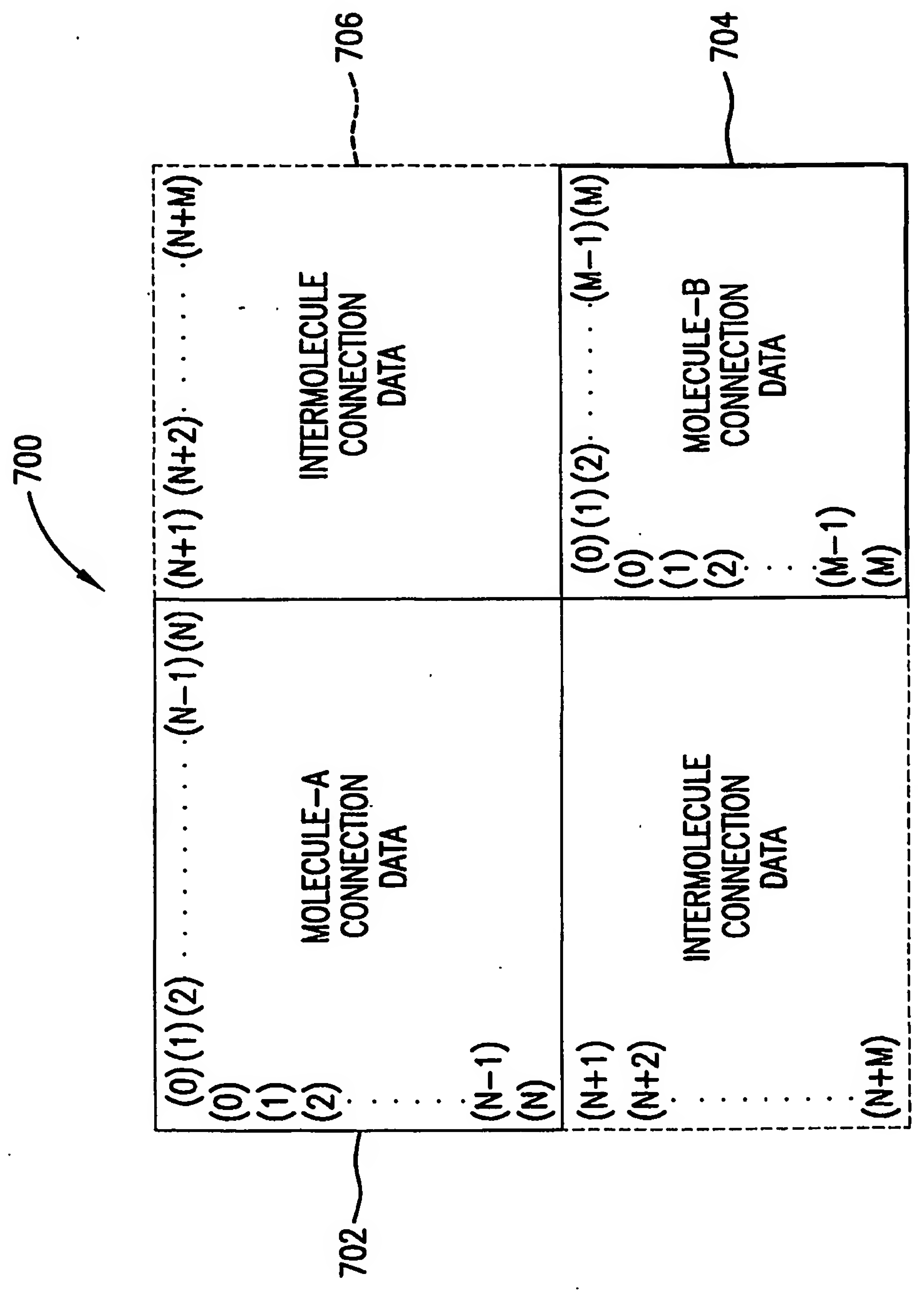

700
706
704
702
(0)(1)(2) . . . . (N−1)(N)
(0)
(1)
(2) . . . . . . (N−1)
(N)
MOLECULE–A CONNECTION DATA
(N+1) (N+2) . . . . . . (N+M)
INTERMOLECULE CONNECTION DATA
(0)(1)(2) . . . . . (M−1)(M)
(0)
(1)
(2) . . . . (M−1)
(M)
MOLECULE–B CONNECTION DATA
(N+1)
(N+2) . . . . (N+M)
INTERMOLECULE CONNECTION DATA

FIG. 7

```
PROC REDUCTIVE_AMINATION {LIBRARY AMINES ALDEHYDES}
{
  DEFINE REAGENT AMINE AS "[C,c][NH2]" OR "C[NH]C" AND NOT
"C(=O)[NH,NH2]";
  DEFINE REAGENT ALDEHYDE AS "C[CH]=O";
  DEFINE PRODUCT p;
  p ADD AMINE FROM $AMINES;
  p ADD ALDEHYDE FROM $ALDEHYDES;
  p INSERT BOND ALDEHYDE:1 AMINE:1;
  p REMOVE ATOM ALDEHYDE:2;
  ENUMERATE p AS $LIBRARY;
}
```

FIG. 8

```
<PRODUCT> ADD <REAGENT> FROM <SOURCE>
<PRODUCT> INSERT BOND <r1:a1> <r2:a2> [<ORDER>] [<STEREO>]
<PRODUCT> REMOVE ATOM <r1:a1>
<PRODUCT> REMOVE BOND <r1:a1> <r1:a2>
<PRODUCT> REMOVE ATTACHMENT <r1:a1>
<PRODUCT> REMOVE FRAGMENT <r1:a1> <r1:a2>
<PRODUCT> SET ATOM <r1:a1> CHARGE|RADICAL <VALUE>
<PRODUCT> SET BOND <r1:a1> <r1:a2> <ORDER> [<STEREO>]
<PRODUCT> SET ATOM <r1:a1> CONFIGURATION <r1:a2> <r1:a3>
<r1:a4> <r1:a5> UP|DOWN
```

FIG. 9

```
PROC CYCLO_ADDITION {LIB OLEFINS BROMINATING_AGENTS} {
  DEFINE REAGENT OLEFIN AS "C=C";
  DEFINE REAGENT BRAGENT AS "C([Br])([Br])[Br]";
  DEFINE PRODUCT p;
  p ADD OLEFIN FROM $OLEFINS;
  p ADD BRAGENT FROM $BROMINATING_AGENTS;
  p INSERT BOND OLEFIN:0 BRAGENT:0;
  p INSERT BOND OLEFIN:1 BRAGENT:0;
  p REMOVE ATOM BRAGNET:3;
  p SET BOND OLEFIN:0 OLEFIN:1 SINGLE  SYN_PRODUCT;
  ENUMERATE p AS $LIB;
};
```

FIG. 10

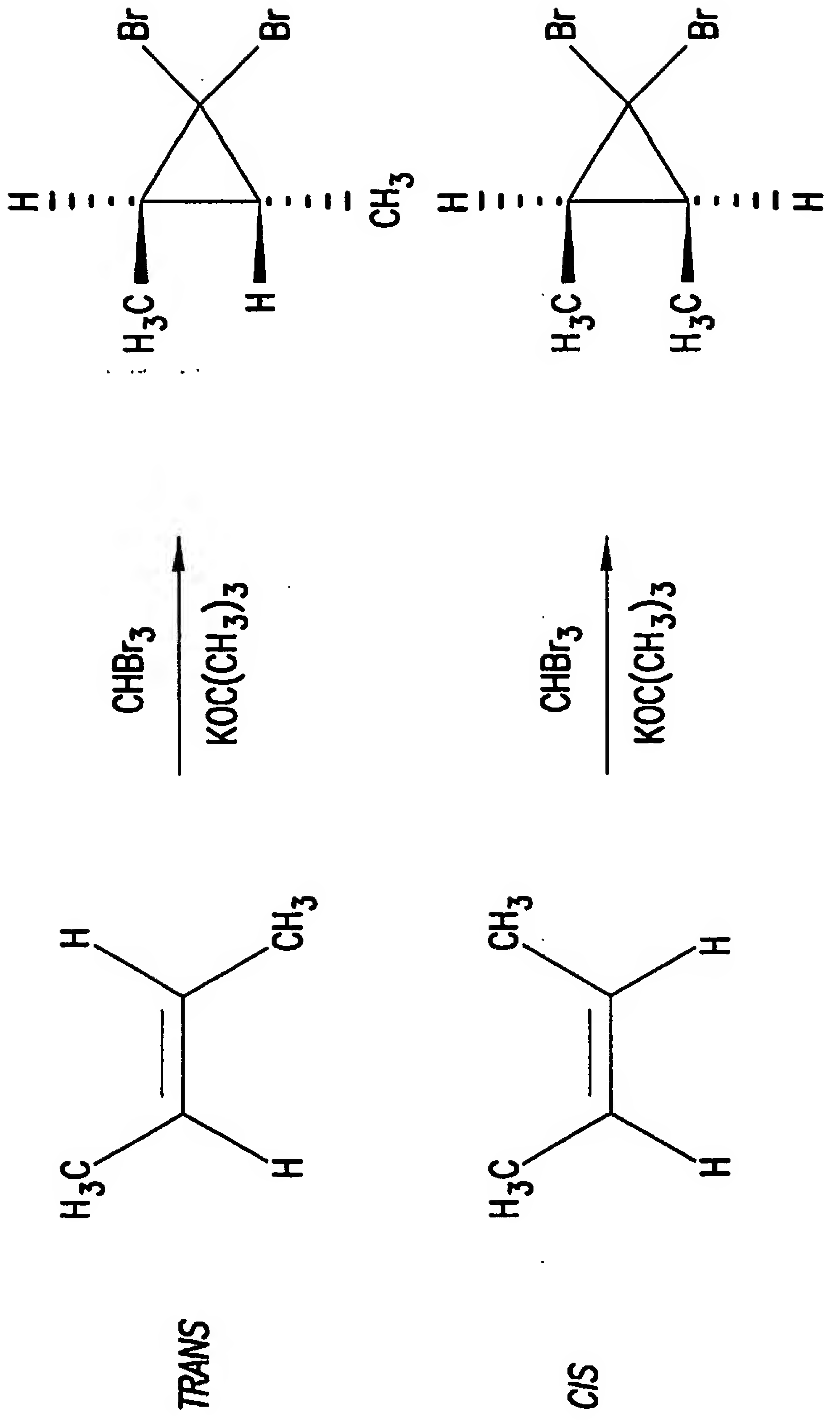


FIG. 11

```
PROC ANTI_ELIMINATION {LIB r1s} {
   DEFINE REAGENT r1 AS "[CH]C[Br]";
   DEFINE PRODUCT p;
   p ADD r1 FROM $r1s;
   p REMOVE ATOM r1:2;
   p SET BOND r1:0 r1:1 DOUBLE ANTI_PRODUCT;
   ENUMERATE p AS $LIB;
   };
```

FIG. 12

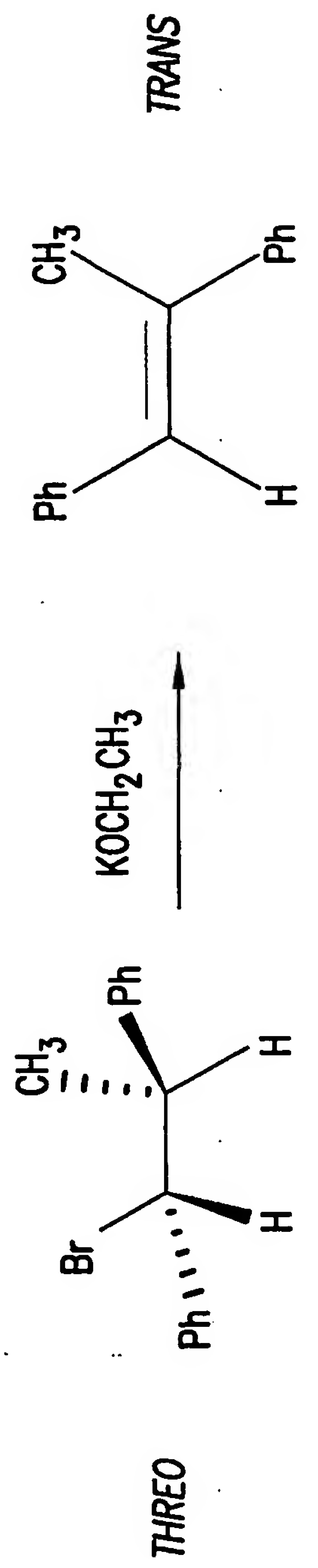


FIG. 13

```
PROC SN_INVERSION {LIB r1s r2s} {
  DEFINE r1 AS "[C&X4]O";
  DEFINE r2 AS "C(=O)O";
  DEFINE PRODUCT p;
  p ADD r1 FROM $r1s;
  p ADD r2 FROM $r2s;
  p REMOVE FRAGMENT r1:0 r1:1;
  p INSERT BOND r1:0 r2:2 SINGLE;
  p SET ATOM r1:0 CONFIGURATION INVERSE;
  ENUMERATE p AS $LIB;
 };
```

FIG. 14

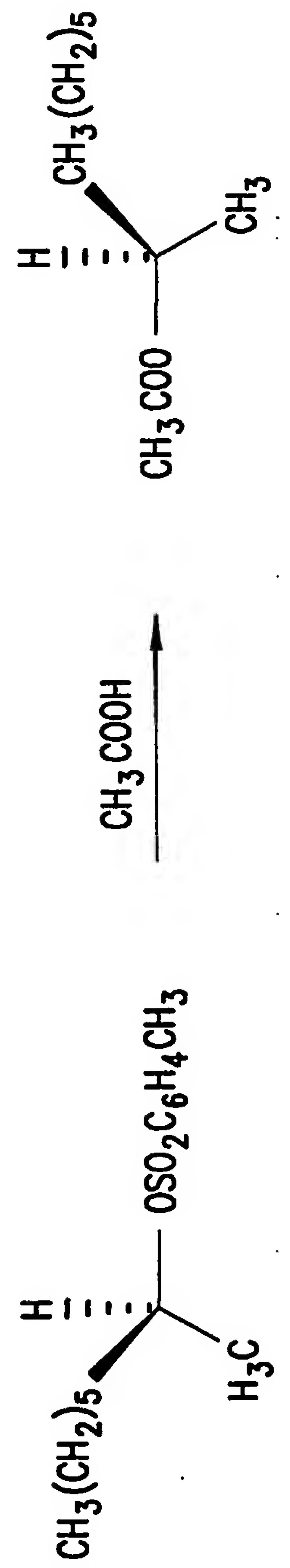


FIG. 15

```
PROC DIELS_ALDER {LIB r1s r2s} {
  DEFINE REAGENT DIENE AS "C1C=CC=C1";
  DEFINE REAGENT DIENOPHILE AS "*C=C*";
  DEFINE PRODUCT p;
  p ADD DIENE FROM $r1s;
  p ADD DIENOPHILE FROM $r2s;
  p INSERT BOND DIENE:1 DIENOPHILE:1;
  p INSERT BOND DIENE:4 DIENOPHILE:2;
  p SET BOND DIENE:1 DIENE:2 SINGLE;
  p SET BOND DIENE:3 DIENE:4 SINGLE;
  p SET BOND DIENE:2 DIENE:3 DOUBLE;
  p SET BOND DIENOPHILE:1 DIENOPHILE:2 SINGLE;
  p SET ATOM DIENOPHILE:1 CONFIGURATION DIENOPHILE:0 DIENOPHILE:2 DIENE:1 H UP;
  p SET ATOM DIENOPHILE:2 CONFIGURATION DIENE:4 DIENOPHILE:1 DIENOPHILE:3 H UP;
  p SET ATOM DIENE:1 CONFIGURATION DIENE:2 DIENE:0 DIENOPHILE:1 H DOWN;
  p SET ATOM DIENE:4 CONFIGURATION DIENOPHILE:2 DIENE:0 DIENE:3 H DOWN;
  ENUMERATE LIBRARY AS $LIB;
  };
```

FIG. 16

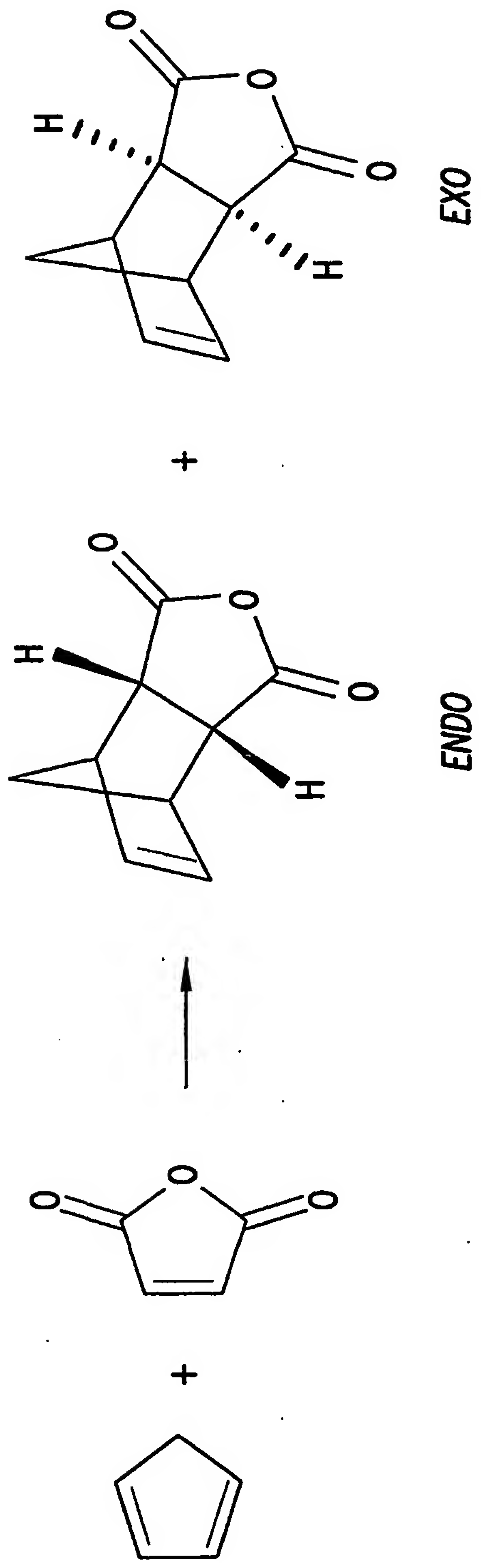


FIG. 17

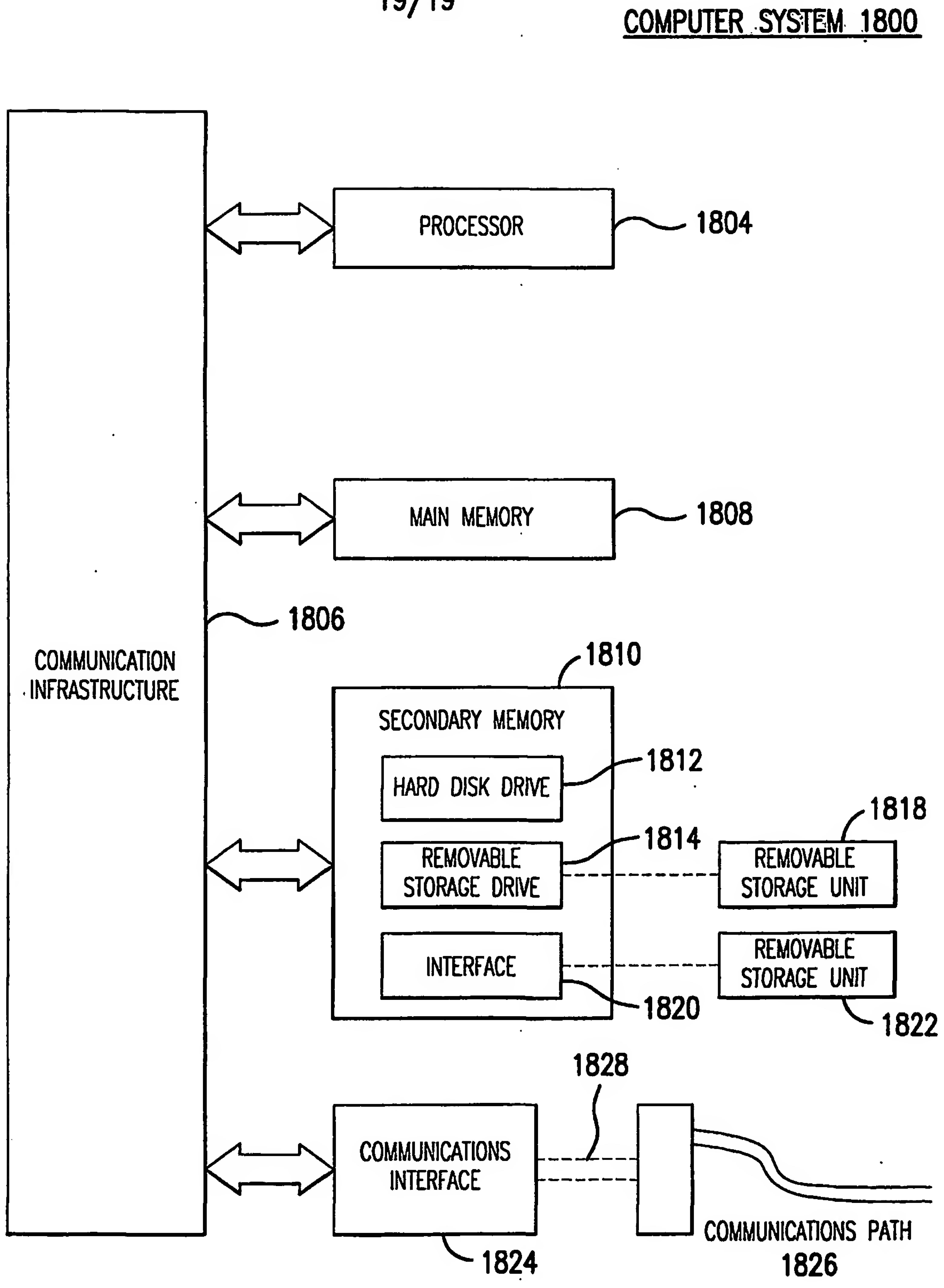

COMPUTER SYSTEM 1800
PROCESSOR
1804
MAIN MEMORY
1808
1806
COMMUNICATION INFRASTRUCTURE
1810
SECONDARY MEMORY
HARD DISK DRIVE
1812
REMOVABLE STORAGE DRIVE
1814
REMOVABLE STORAGE UNIT
1818
INTERFACE
1820
REMOVABLE STORAGE UNIT
1822
1828
COMMUNICATIONS INTERFACE
1824
COMMUNICATIONS PATH
1826

FIG. 18